CLAYTON STATE UNIVERSITY

Quality testing of data from photon-based hardware random number generator

Dmitriy Beznosko*, Keith Driscoll, Fernando Guadarrama, Alexander Iakovlev, Steven Mai, Nikolas Thornton
ICHEP 2024 *presenter, dmitriybeznosko@clayton.edu

COSMIC RAYS AND PARTICLES GROUP

Introduction

HRNG, or the Hardware random number generators, nowadays are widely used both in the computer world for the security purposes and by many sciences for the models and simulations as a source of the high-quality randomness . HRNGs that exist currently are either expensive or slow with questionable random data quality in the long-term stability tests. This study applies the capabilities of an already established HRNG design [1, 2] and introduces new tests for the randomness of a data set of the HRNG based on the low-number photon absorption by a detector (a photo-multiplier tube of a silicon-based photodetector) [3] that can provide a large volume of high-quality random numbers. The results of testing of quality of the generator output are presented.

Pseudo-random number generators

PRNGs are typically based on hard-to-predict algorithms that are based on the seed – a number that PRNGs use to derive their randomness from unpredictable or difficult to replicate factors, such as the user’s mouse movement or the machine’s uptime.

Pros:

- Easily replicable results
- No external hardware necessary

Cons:

- Easily replicable results
- Not random enough for many use cases

Photon-based simple quantum solution to random number generation

Photomultiplier Tube

A photomultiplier tube (PMT) detects photons from the LED that is powered by the pulse from an external generator that also triggers the data acquisition (DAQ). The main downside of this setup is that it requires a black-box for the PMT and the LED to be cased in and uses high voltage of ~ 2000V. In Figure 1, the setup is using a Hamamatsu [4] R7723 PMT and a blue LED.

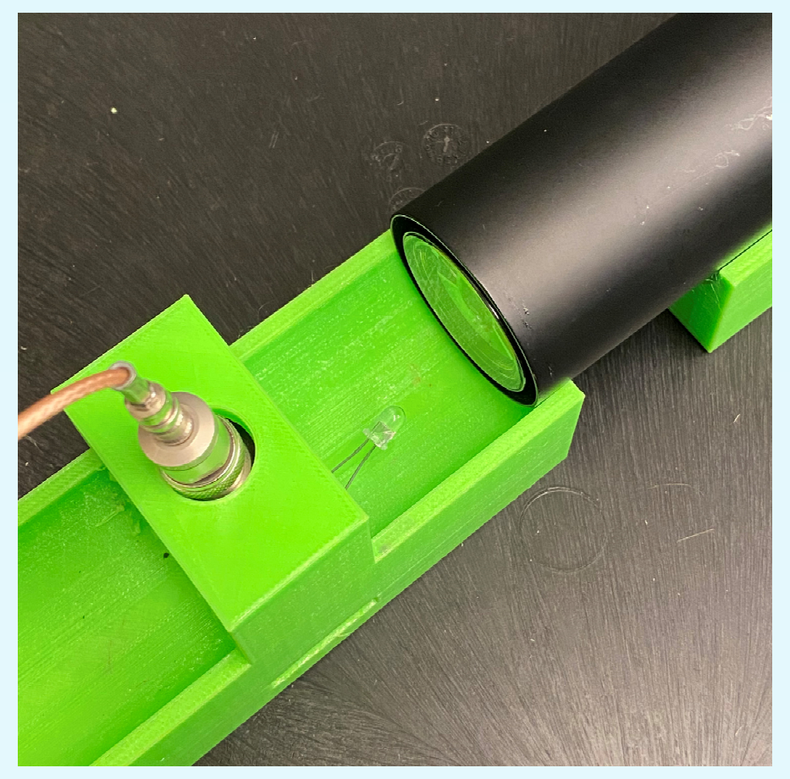

Figure 1. *The PMT HRNG setup*

Multi-Pixel Photon Counter

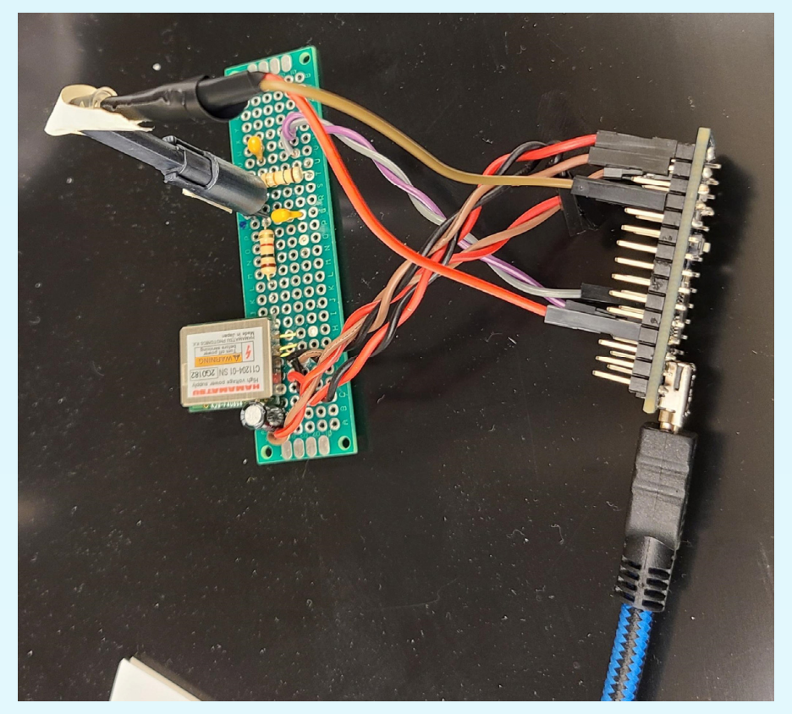

Figure 2. *The MPPC HRNG setup*

A Multi-Pixel Photon Counter (MPPC) from Hamamatsu [4] is connected to an Arduino and detects photons in sync with an LED pulse from an internal trigger. The LED is powered by the Arduino’s default Pulse Width Modulation (PWM). The sensor is in a dark enclosure tube.

This prototype has shown enormous potential despite the current setup generating only about 4000 random amplitudes per second due to the slow Arduino board.

Testing Methods

The methods used for testing the data quality are described. The collected data is tested for the quality of the randomness of the bits obtained. Simplest test includes finding the Arithmetic Mean, Standard Deviation and Entropy of the dataset – the AMSDET. Note – entropy test is described separately.

Monte Carlo Pi Estimation Test

The Monte Carlo Pi Estimation Test (MCPET) is done by randomly selecting points within a square and counting each time a point falls within the quarter-cross-section of a circle, and then dividing that total by the total number of points everywhere, pi can be estimated. This test relies on having enough coordinates produced from data to accurately estimate pi. For analysis, pi is calculated from the data and by using Python’s Random library with the same number of coordinates as the data. Comparing both helps determine whether a poor pi value is due to limited amount of data or poor-quality data.

Fractional Line Symmetry Test

The Fractional Line Symmetry Test, or FLST [5], is a test ***developed specifically for this project*** that compares how frequently bits appear when stacked and visualized. This test is sensitive to poor quality of the data sample that otherwise passes simple tests like the standard deviation and average.

Horizontal/Vertical FLST

HV-FLST compares how frequently bits appear back-to-back horizontally and vertically when stacked and visualized into a 2D image. The folding done to stack and visualize the data is inherently random to the data length itself. Once the linear data is turned into a 2D image, the number of “lines” (back-to-back bits of some length) found horizontally as in Figure 3 (left), and vertically as in Figure 3 (middle), should be the same in the ideal sample. The prediction is the same for the horizontal and vertical, so comparison with the predicted value also gives cross-comparison between vertical and horizontal.

Diagonal FLST

The D-FLST is an implementation of the HV-FLST, except the bits data is first turned into a square matrix and traversed diagonally. The bitstream is then traversed starting in the top left corner, moving up and to the right one cell, wrapping back to the bottom left once the edge of matrix is reached. This traversal path is shown in Figure 3 (right), where the numbering attached to the start of the green arrows describes the order that the individual diagonals are traversed. For all FLST variations, results are compared with the theoretical predictions as given by Equation 1. n is line length.

$$Lines\# = \frac{n+1}{(2^{n+1})n} \times bitstream\ length \quad (1)$$

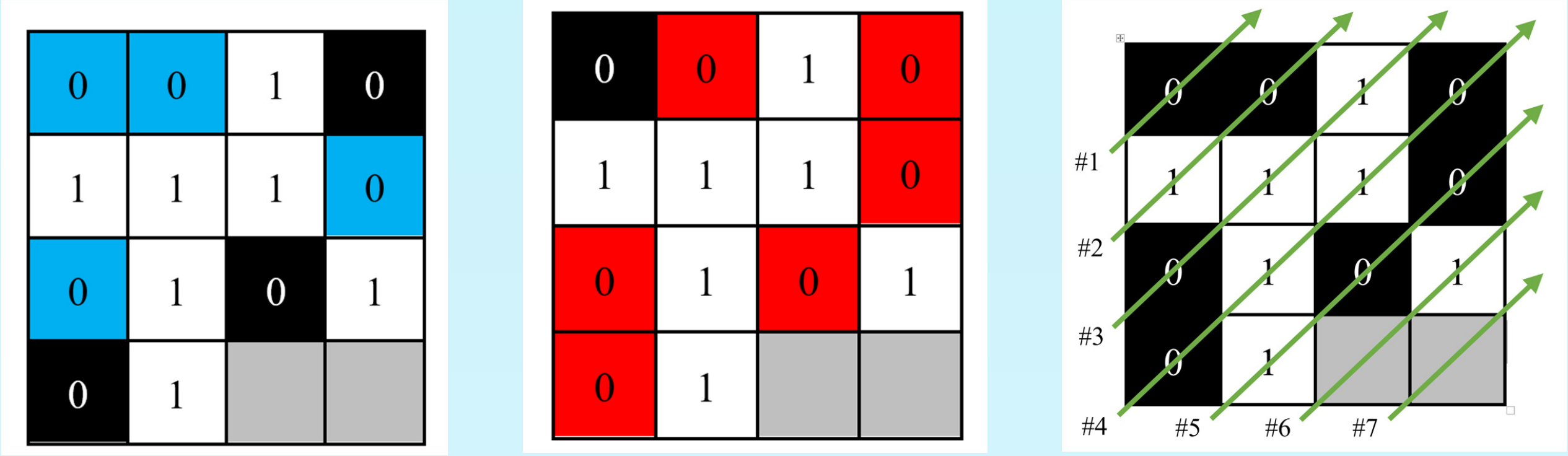

Figure 3. **Left:** *Horizontal lines found* **Center:** *Vertical lines found* **Right:** *Diagonal lines found.*

Entropy Test

The Entropy Test (part of AMSDET) evaluates the overall *balance* of a given data set, i.e., it provides an initial overview of the ratio of 1’s to 0’s. The higher the entropy value the more balance the set has. The entropy test was implemented by taking a bitstream and calculating its entropy value by using Shannon’s entropy formula [6], as shown in Equation 2.

$$-(P(0)\cdot log_2(P(0)) + P(1)\cdot log_2(P(1))) \quad (2)$$

where $P(0)$ is the probability of 0’s appearing in the data and $P(1)$ is the probability of 1’s appearing. The value of $P(0)$ was determined by taking the total number of 0’s in the bitstream and dividing it by the length of the bitstream. The same process was used to find the value of $P(1)$. The result yields an entropy value for the dataset, which determines the balance of 1’s and 0’s in a data set, where repetitive data has an entropy value close to 0 and more balanced data has an entropy value close to 1. It is important to note that the balance of 1’s and 0’s is an attribute important to random data but does not necessarily determine whether the data is random.

Initial Processing of Amplitudes

Before the data can be further processed, it needs to be turned into a bitstream, so a set procedure should exist to turn the amplitudes from the photosensor into the 1s and 0s. Figure 4 shows the pseudo-code used for this purpose – if the amplitude is even, it is assigned a bit value of 1, and 0 for the odd ones. Additionally, a known ‘poor’ data-set is created by using the procedure in Figure 5, where if the next amplitude is higher than previous, 1 is assigned, and 0 for lower. This process creates subtle asymmetry, and we used this dataset for the validation of the data quality testing methods used.

```
if amp[i] % 2 == 0: bitstream +=1
else: bitstream += 0
```

Figure 4. *Even/Odd amplitude processing pseudo-code*

```
if amp[i] > amp[i-1]: bitstream += 1
if amp[i] < amp[i-1]: bitstream += 0
else: skip
```

Figure 5. *High/Low amplitude processing pseudo-code*

Iterated Von Neumann Extractor

IVNE procedure was initially designed to equalize probabilities for the ‘unfair’ coin. For the coin:

- If $P(heads) = x$ then $P(tails) = 1 - x$
- In general: $x \neq 1 - x$
- Look at pairs: heads + tails (ht) and tails + heads (th).
- Probability for a pair is a product of individual ones.
- $P(ht) = x(1-x)$ and $P(th) = (1-x)x$
- $P(ht) = P(th)$, QED

The same process can be applied to balance the bitstream by folding it in two and counting 10 and 01 pairs and discarding 11 and 00 pairs. The 10 is replaced by 1, 01 by 0 and a new bitstream is formed. Process is iterated over the discarded data. This ‘extracts’ all the randomness from the data set.

Figure 6. Coin example: Heads and Tails

Experimental Results

Table 1. Datasets that were used.

Dataset	Processing Method	Raw Size, bits	Post-IVNE Size, bits
Small PMT Bitstream	Even/Odd	290000	282888
	High/Low	289050	269145
Small MPPC Bitstream	Even/Odd	290000	282845
	High/Low	266251	245024
Large MPPC Bitstream	Even/Odd	4080134	4034055
	High/Low	3752644	3488717

Table 2. AMSDET results for the PMT data using Even/Odd processing.

Dataset, Even/Odd	IVNE	Average	Standard deviation	Entropy
Small PMT Bitstream	✗	0.4994620690	0.4999997106	0.9999999072
	✓	0.4994732898	0.4999997226	0.9999999111

Table 3. AMSDET results, PMT data, using High/Low processing (bad data)

Dataset, High/Low	IVNE	Average	Standard deviation	Entropy
Small PMT Bitstream	✗	0.4999238886	0.4999999942	0.9999999981
	✓	0.4987237363	0.4999983711	0.9999994778

Simple methods of AMSDET can’t identify the known deficiency of high/low processing. Similar results were obtained for the MPPC datasets and are not shown here.

Monte Carlo Pi Estimation Test

Table 4. MCPET results for the PMT data using Even/Odd processing.

Dataset, Even/Odd	IVNE	Estimated Pi	Python’s Random() Pi
Small PMT Bitstream	✗	3.13644	3.15807
	✓	3.14502	3.13982

Table 5. MCPET results, PMT data, using High/Low processing (bad data)

Dataset, High/Low	IVNE	Estimated Pi	Python’s Random() Pi
Small PMT Bitstream	✗	3.45441	3.16258
	✓	3.09256	3.16390

Table 6. MCPET results for the MPPC data using Even/Odd processing.

Dataset, Even/Odd	IVNE	Estimated Pi	Python’s Random() Pi
Small MPPC Bitstream	✗	3.14196	3.13688
	✓	3.13311	3.13469
Large MPPC Bitstream	✗	3.16522	3.14211
	✓	3.13517	3.14020

Table 7. MCPET results, PMT data, using High/Low processing (bad data)

Dataset, High/Low	IVNE	Estimated Pi	Python’s Random() Pi
Small MPPC Bitstream	✗	3.48942	3.12236
	✓	3.07327	3.13830
Large MPPC Bitstream	✗	3.48701	3.14268
	✓	3.08231	3.14122

MCPET is quite sensitive and can clearly identify the known deficiency of high/low processing. Even application of INVE procedure doesn’t fix the bad dataset deficiencies that affect MCPET.

The FLST Results

Detect length of n=**4**.

The FLST Visual Example

Estimated line count for 290,000 bits: **11328**

Even/Odd

```
Horizontal lines:
PRE-NEUMANN:  11337.5
POST-NEUMANN: 10990.0

Vertical lines:
PRE-NEUMANN:  11291.75
POST-NEUMANN: 11006.25
```

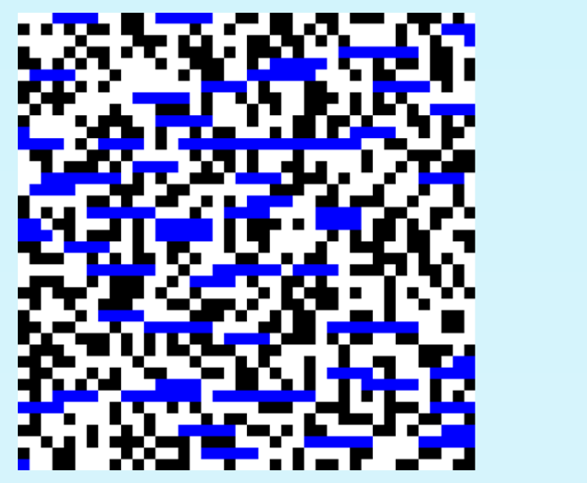

Figure 7. *Even/odd pre-Neumann PMT data horizontal lines*

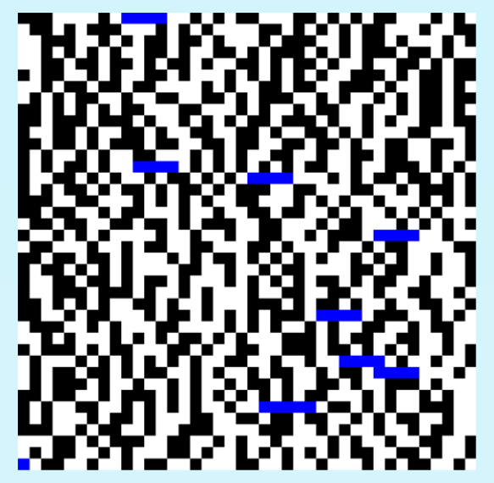

Figure 8. *High/low pre-Neumann PMT data horizontal lines*

High/Low
(bad data)

```
Horizontal lines:
PRE-NEUMANN:  2042.25
POST-NEUMANN: 11981.25

Vertical lines:
PRE-NEUMANN:  11359.75
POST-NEUMANN: 10495.25
```

Table 8. FLST results for the PMT data using Even/Odd processing.

Dataset, Even/Odd	IVNE	Horizontal % Discrepancy	Vertical % Discrepancy	Diagonal % Discrepancy
Small PMT Bitstream	✗	0.08276%	0.3211%	0.1556%
	✓	0.5458%	0.3987%	0.9270%

Table 9. FLST results, PMT data, using High/Low processing (bad data)

Dataset, High/Low	IVNE	Horizontal % Discrepancy	Vertical % Discrepancy	Diagonal % Discrepancy
Small PMT Bitstream	✗	81.91%	0.6088%	0.1593%
	✓	13.96%	0.1734%	0.2827%

It is very clear that these simple methods find the deficiency of high/low conversion method, specifically the MCPE. IVNE fails to improve the dataset and even makes is a bit worse in some cases.

Table 10. FLST results for the MPPC data using Even/Odd processing.

Dataset, Even/Odd	IVNE	Horizontal % Discrepancy	Vertical % Discrepancy	Diagonal % Discrepancy
Small MPPC Bitstream	✗	2.215%	3.642%	2.616%
	✓	1.076%	0.7140%	0.4108%
Large MPPC Bitstream	✗	3.535%	3.671%	3.527%
	✓	0.4271%	0.1133%	0.4147%

Table 11. FLST results, MPPC data, using High/Low processing (bad data)

Dataset, High/Low	IVNE	Horizontal % Discrepancy	Vertical % Discrepancy	Diagonal % Discrepancy
Small MPPC Bitstream	✗	87.85%	0.8300%	0.6833%
	✓	13.84%	1.481%	1.951%
Large MPPC Bitstream	✗	88.26%	0.2699%	0.4382%
	✓	14.56%	0.1178%	0.1871%

The MPPC setup requires fine-tuning like finding the right biasing voltage and amount of light that illuminates the sensor, and this data was obtained prior to such work that is currently in progress.

The FLST should show the symmetric results between the vertical and horizontal lines. By using different detect length, detection of patterns that indicate lower data quality is possible. This is specifically clear when comparing the number of lines for the data obtained high/low method before IVNE procedure is applied. This is expected, and the intended low-quality of High/Low data is clearly caught by FLST even when compared to MCPET.

Conclusion

- The photon-based solutions explored in this research have shown great potential for being high-speed sources of high-quality random number generation.
- It was found as shown in Experimental Results that the current best way to process data into the bitstream is the Even/Edd method. The PMT appears to outperform the MPPC in the MCPE Test, but the MPPC setup is still in its prototype stage and needs tuning.
- By use of the FLST, it was detected that the high/low method of processing is far worse than the even/odd method as expected. The FLST successfully revealed this together with MCPET, even though the high/low algorithm otherwise passed the AMSDET well.
- Current objectives for the future of this project are to sync the MPPC + Arduino with an LED that is voltage controlled through means besides PWM. A proper enclosure for the board and sensor is also necessary, as well as a faster board to increase the number of amplitudes per second.
- Additionally, plans include completing the data collection of the large samples and testing them with the methods described in this work. The results will be compared with other known tests such as “Die hard”, TestU01 and similar.

References

[1] D. Beznosko et al., "Random Number Hardware Generator Using Geiger-Mode Avalanche Photo Detector", e -Print: 1501.05521, DOI: 10.22323/1.252.0049, PoS PhotoDet2015 (2016), 049
[2] A. Iakovlev et. al, “Random Number Hardware Generator Using Geiger-Mode Avalanche Photo Detector”, arXiv preprint arXiv:1501.05521, 2015/1/22 https://doi.org/10.48550/arXiv.1501.05521
[3] Dmitriy Beznosko, Keith Driscoll, Fernando Guadarrama, Steven Mai, Nikolas Thornton. “Data Analysis Methods Preliminaries for a Photon-based Hardware Random Number Generator”, arXiv:2404.09395 [cs.CR] 15 Apr 2024
[4] Hamamatsu Photonics K.K., 325-6, Sunayama-cho, Naka-ku, Hamamatsu City, Shizuoka Pref., 430-8587, Japan
[5] Dmitriy Beznosko, Keith Driscoll, Fernando Guadarrama, Steven Mai, Nikolas Thornton, “Preliminaries of a Photon-based Hardware Random Number Generator Design and Data Analysis Methods”, 3rd Annual College of STEM Symposium, PROC(03ACSS2024)001 https://sos.clayton.edu/proceedings/003/PROC(03ACSS2024)001.pdf
[6] SHANNON, C.E. “A Mathematical Theory of Communication.” Harvard Mathematics Department, Accessed 16 April 2024. https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.